Developing Zynq Software with Xilinx SDK

Lab 7

Boot from Flash

August 2016
Version 08

Lab 7 Overview

With the FSBL in place, we are now ready to create a boot image and boot one of our applications from non-volatile memory. A complete boot-up requires at least three things:

1. FSBL
2. Bitstream
3. Application

This Lab will show you how to combine these pieces together to create the boot images for both QSPI and SD Card. You will then be able to experiment with a true embedded, non-tethered boot.

Lab 7 Objectives

When you have completed Lab 7, you will know:

- How to create a boot image
- Write and boot from QSPI
- Write and boot from SD Card

Experiment 1: Create the Boot Image

The first step is to create the non-volatile boot images. Our hardware has two non-volatile bootable sources, QSPI and microSD/SD Card.

Experiment 1 General Instruction:

Change the configuration for the Peripheral Test to Release. Create a boot image for both QSPI and SD Card.

Experiment 1 Step-by-Step Instructions:

We first need to decide which application that we will bootload. The important consideration here is that you do not want to choose an application that will compete for the same memory resource as the FSBL. Recall from Lab 6 that the FSBL is ~140K, targeted at the on-chip RAM_0.

Since the Test_Memory application runs from the same memory, this won't work. We may be able to re-design the linker script to split this application across the remaining RAM0 and also RAM1, but that is beyond the scope of this experiment.

The Hello_Zynq application is a good candidate, except recall that we targeted this application to on-chip RAM. We would need to modify the linker script again, or modify the linker to put this application in the remaining RAM0 and also RAM1.

The Test Peripheral application is targeted at DDR. Since this application is compatible immediately with the FSBL, we will use it.

1. Normally Build Configurations would be set to Active Release, however the Peripheral application currently has an Optimization issue. A Change Request has been made under CR-956814. For now we will use **Debug Build Configuration.**

2. In the main SDK menu, select **Project → Build All** to force the new configurations to build.

3. In SDK, select the Test_Peripherals application.

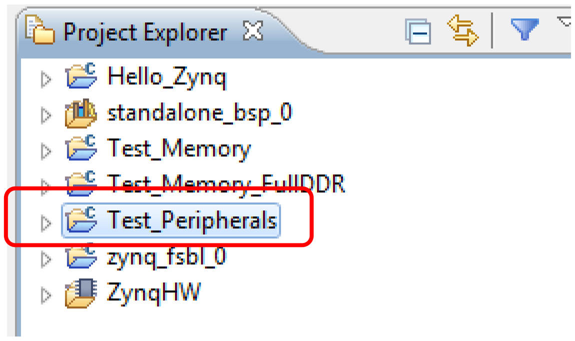
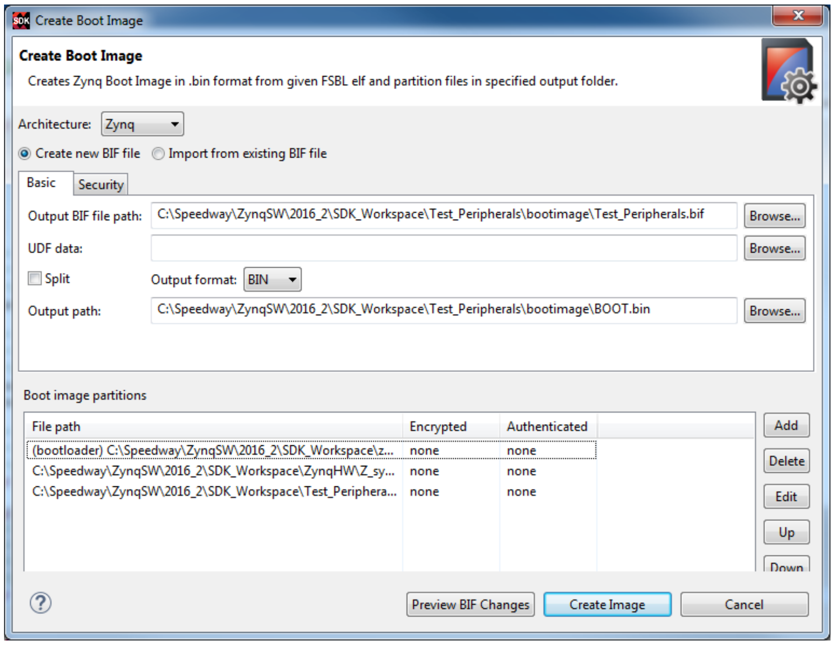


Figure 1 – Select Application to Boot

4. Select **Xilinx Tools → Create Boot Image**.



Create Boot Image

Creates Zynq Boot Image in .bin format from given FSBL elf and partition files in specified output folder.

Architecture: Zynq

(● Create new BIF file) ○ Import from existing BIF file

Output BIF file path: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\bootimage\Test_Peripherals.bif

UDF data:

☐ Split   Output format: BIN

Output path: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\bootimage\BOOT.bin

Boot image partitions

File path	Encrypted	Authenticated
(bootloader) C:\Speedway\ZynqSW\2016_2\SDK_Workspace\z...	none	none
C:\Speedway\ZynqSW\2016_2\SDK_Workspace\ZynqHW\Z_sy...	none	none
C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Periphera...	none	none

Figure 2 – Initial Create Zynq Boot Image Dialog

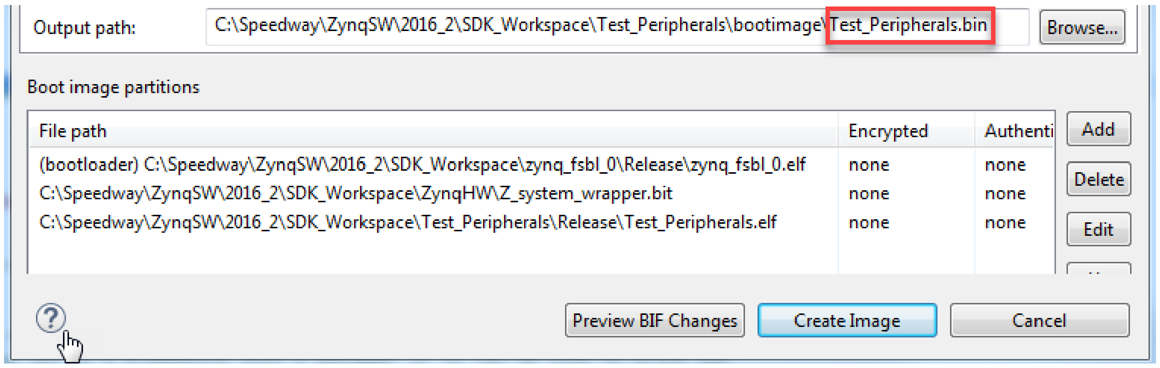
5. If your dialog is not pre-populated with files as shown above, then your Test_Peripherals application may have an issue. Click **Cancel**, then right-click on Test_Peripherals and select **Clean Project**.

6. In the dialog, leave the radio button selected for *Create a new BIF file*. BIF stands for Boot Image Format. The BIF is the input file into Bootgen that lists the partitions (bitstream, software) which Bootgen is to include in the image. The BIF also includes attributes for the partitions. Partition attributes allow the user to specify if the partition is to be encrypted and/or authenticated.

7. We will not be using Authentication or Encryption, so these checkboxes should be left unchecked.

The *Boot Image Partitions* is prepopulated with the FSBL and Application ELF ***Release*** images as well as the bitstream. SDK 2016.2 has correctly pulled in the ELFs for our active configurations.

8. The *Output Path* area actually determines two things, although this is not obvious. Of course, it sets the output file name. The second thing that it determines is whether it creates an image for the SD Card (bin) or the QSPI (mcs). By default, it is set to bin, so we will create that one first. The directory location of `Test_Peripherals\bootimage` is good. Change the output file name to **Test_Peripherals.bin**.


Output path: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\bootimage\Test_Peripherals.bin

Boot image partitions

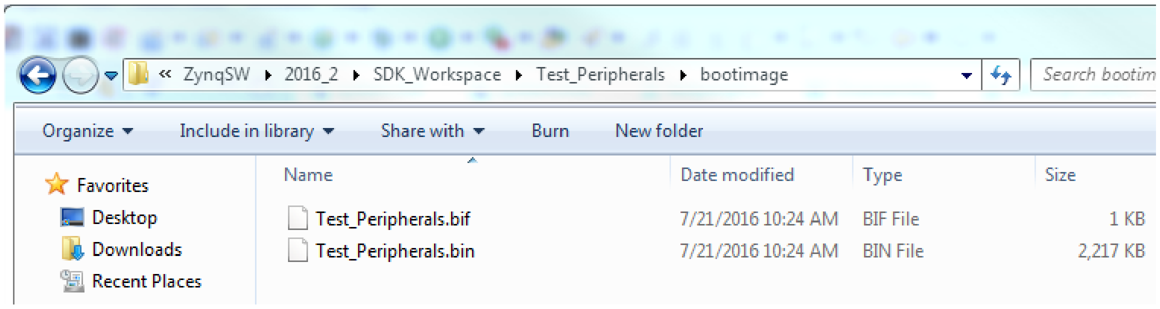
File path	Encrypted	Authenti
(bootloader) C:\Speedway\ZynqSW\2016_2\SDK_Workspace\zynq_fsbl_0\Release\zynq_fsbl_0.elf	none	none
C:\Speedway\ZynqSW\2016_2\SDK_Workspace\ZynqHW\Z_system_wrapper.bit	none	none
C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\Release\Test_Peripherals.elf	none	none

Figure 3 – Zynq Boot Image Dialog

When complete the dialog should look as above, with the FSBL first, the bitstream second, and the application third

9. Click **Create Image**.

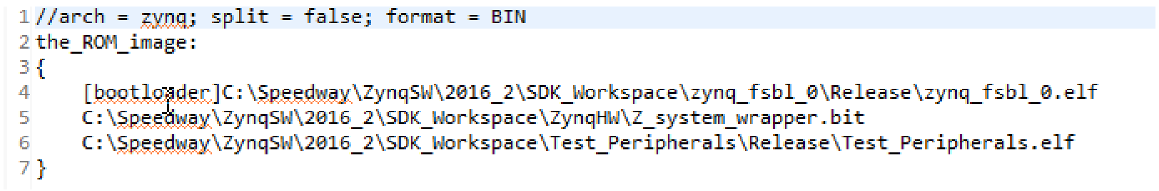
10. Using Windows Explorer, navigate to the application directory, then into the newly created **bootimage** directory. Notice that two files have been created: .bif and .bin. If we were only going to boot from SD Card, this would be enough.



Name	Date modified	Type	Size
Test_Peripherals.bif	7/21/2016 10:24 AM	BIF File	1 KB
Test_Peripherals.bin	7/21/2016 10:24 AM	BIN File	2,217 KB

Figure 4 - bootimage Directory

11. Open the Test_Peripherals.bif file in a text editor. You'll notice that this is a very simple file. The absolute paths to the three files are listed. Close the file.



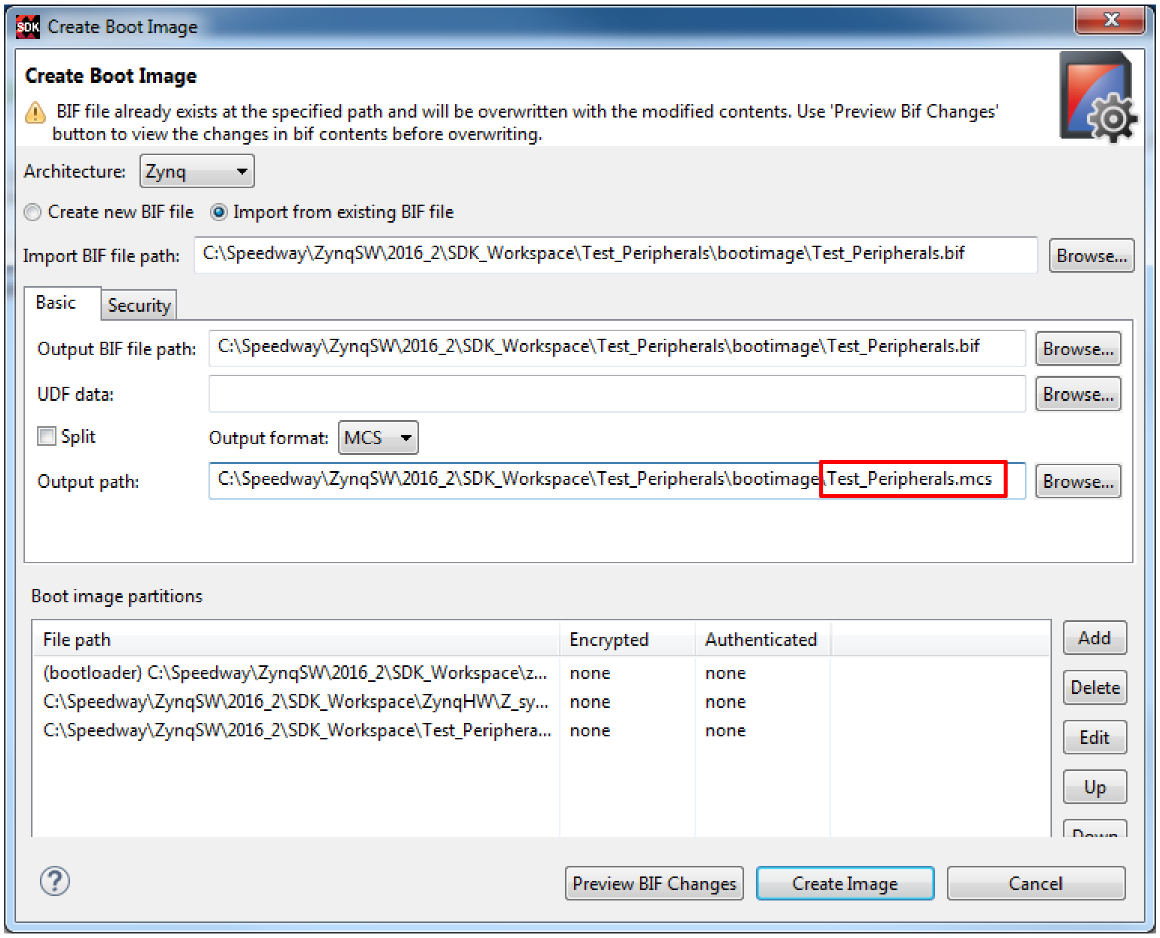
```
//arch = zynq; split = false; format = BIN
the_ROM_image:
{
    [bootloader]C:\Speedway\ZynqSW\2016_2\SDK_Workspace\zynq_fsbl_0\Release\zynq_fsbl_0.elf
    C:\Speedway\ZynqSW\2016_2\SDK_Workspace\ZynqHW\Z_system_wrapper.bit
    C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\Release\Test_Peripherals.elf
}
```

Figure 5 – BIF Contents

12. Launch the **Create Boot Image** utility again.

13. This time, it is OK to import the existing .bif file.

14. For the *Output path*, click **Browse**. Browse to the `SDK_Workspace\Test_Peripherals\bootimage` folder, then change the *Save as type* to **.mcs**. Type in **Test_Peripherals** as the *File name.*


Create Boot Image

Create Boot Image

BIF file already exists at the specified path and will be overwritten with the modified contents. Use 'Preview Bif Changes' button to view the changes in bif contents before overwriting.

Architecture: Zynq

( ) Create new BIF file (●) Import from existing BIF file

Import BIF file path: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\bootimage\Test_Peripherals.bif

Basic | Security

Output BIF file path: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\bootimage\Test_Peripherals.bif

UDF data:

☐ Split — Output format: MCS

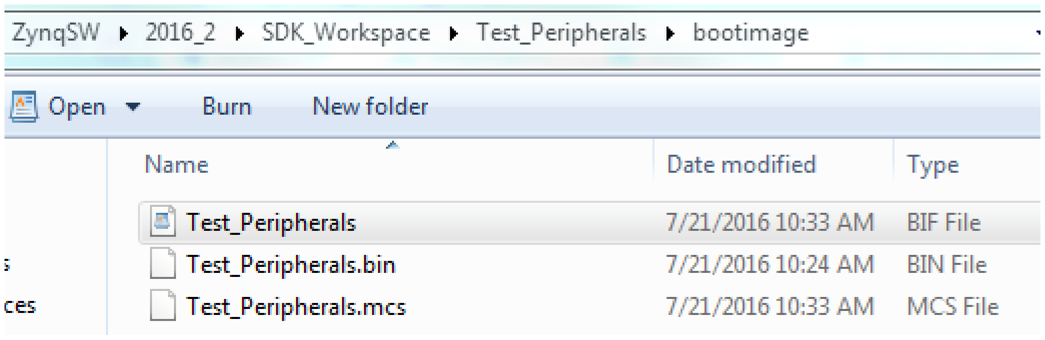
Output path: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\bootimage\Test_Peripherals.mcs

Boot image partitions

File path	Encrypted	Authenticated
(bootloader) C:\Speedway\ZynqSW\2016_2\SDK_Workspace\z...	none	none
C:\Speedway\ZynqSW\2016_2\SDK_Workspace\ZynqHW\Z_sy...	none	none
C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Periphera...	none	none

Figure 6 – Create the MCS Boot Image

15. Click **Create Image** and then check the bootimage folder again to ensure the .mcs has also now been created. Click **OK** to override BIF file.



ZynqSW ▸ 2016_2 ▸ SDK_Workspace ▸ Test_Peripherals ▸ bootimage

Name	Date modified	Type
Test_Peripherals	7/21/2016 10:33 AM	BIF File
Test_Peripherals.bin	7/21/2016 10:24 AM	BIN File
Test_Peripherals.mcs	7/21/2016 10:33 AM	MCS File

Figure 7 – Test Peripherals Boot Images

Experiment 2: Write and boot from QSPI

First, we will program the QSPI with the MCS file from within SDK. Then we will boot the test application.

Experiment 2 General Instruction:

Program the QSPI with the MCS file. Disconnect power then change the MODE jumpers to QSPI. Power on the board and boot from QSPI.

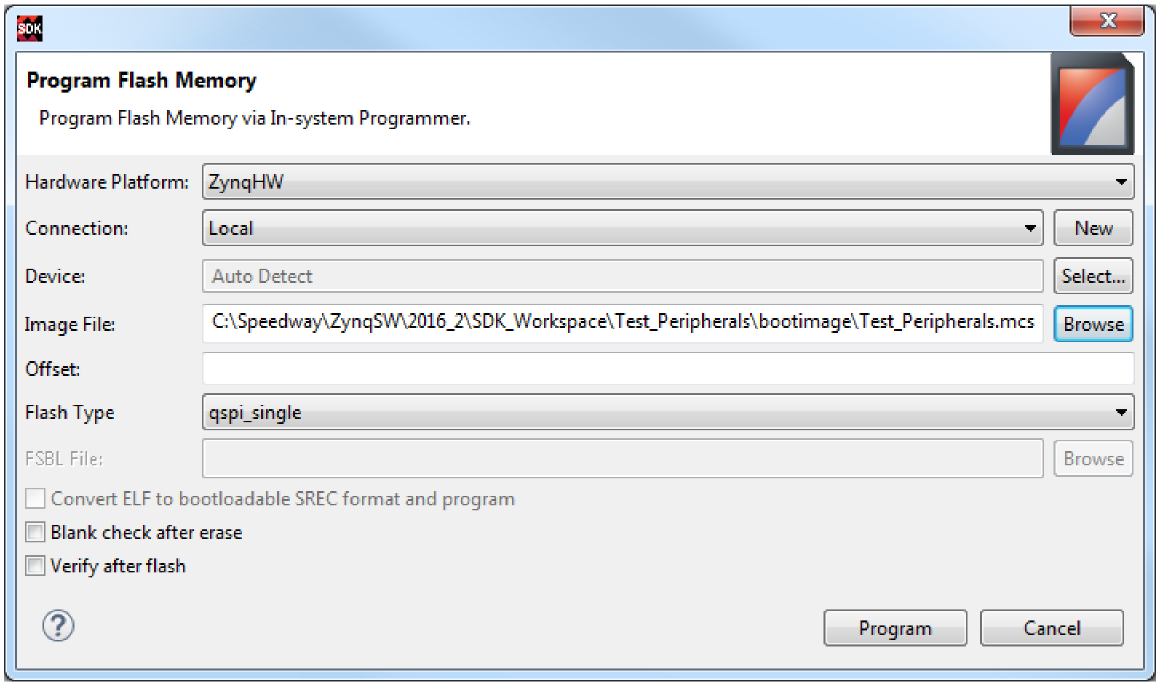
Experiment 2 Step-by-Step Instructions:

1. Set up and turn on your board as before (Cascaded JTAG Mode, USB-UART, and JTAG, turn power on).

2. In SDK, select **Xilinx Tools → Program Flash**.

3. Click the **Browse** button, and browse to the `SDK_Workspace\Test_Peripherals\bootimage` folder then select the `Test_Peripherals.mcs` image. Select it and click **Open**.

4. Click **Program**.



Program Flash Memory

Program Flash Memory via In-system Programmer.

Hardware Platform: ZynqHW

Connection: Local

Device: Auto Detect

Image File: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\Test_Peripherals\bootimage\Test_Peripherals.mcs

Offset:

Flash Type: qspi_single

FSBL File:

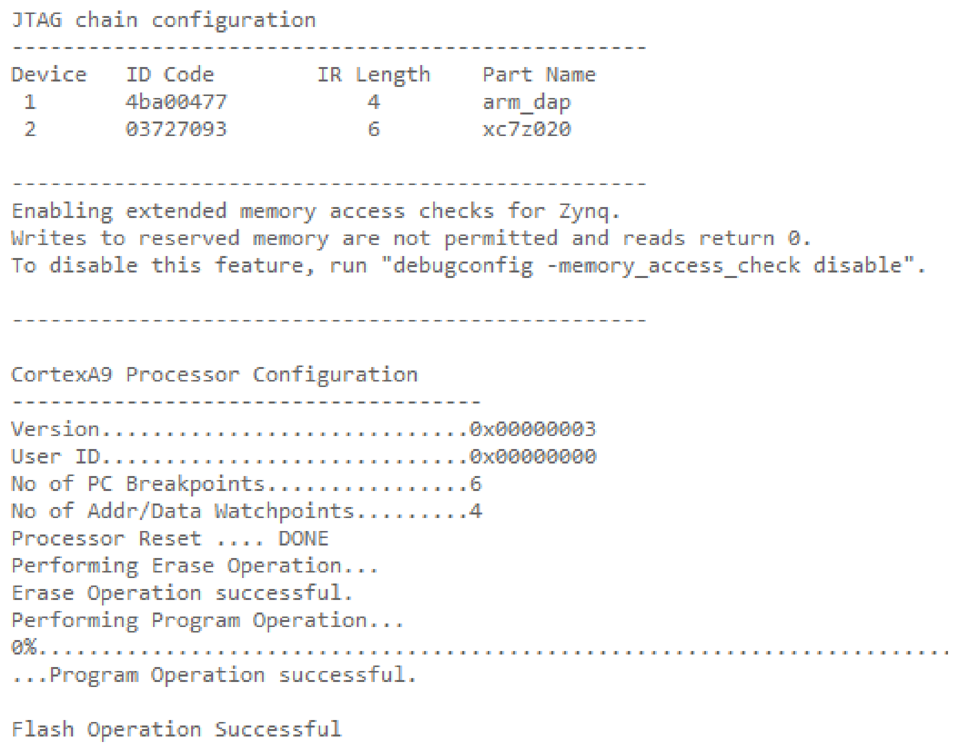
☐ Convert ELF to bootloadable SREC format and program

☐ Blank check after erase

☐ Verify after flash

Figure 8 – Program QSPI Flash Memory

5. The operation should take approximately 1 minute. You should see the following in the Console window.


```
JTAG chain configuration
--------------------------------------------------
Device   ID Code        IR Length    Part Name
 1       4ba00477           4        arm_dap
 2       03727093           6        xc7z020

--------------------------------------------------
Enabling extended memory access checks for Zynq.
Writes to reserved memory are not permitted and reads return 0.
To disable this feature, run "debugconfig -memory_access_check disable".

--------------------------------------------------

CortexA9 Processor Configuration
-------------------------------------
Version.............................0x00000003
User ID.............................0x00000000
No of PC Breakpoints................6
No of Addr/Data Watchpoints.........4
Processor Reset .... DONE
Performing Erase Operation...
Erase Operation successful.
Performing Program Operation...
0%.......................................................................
...Program Operation successful.

Flash Operation Successful
```

Figure 9 – QSPI Programmed Successfully

6. *<ZedBoard & PicoZed Only>* Turn power off.

7. Unplug the USB-UART cable and the JTAG cable.

8. Set the Boot Jumpers to QSPI.

   a. ZedBoard & PicoZed

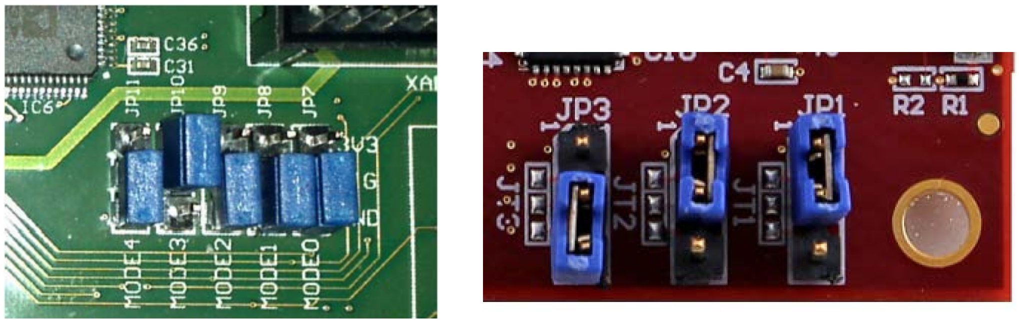


Figure 10 – QSPI Boot Mode: ZedBoard left; MicroZed Right

b. PicoZed

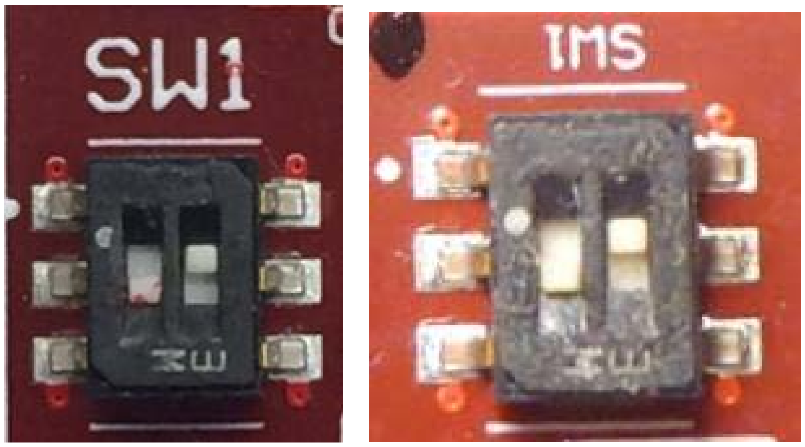
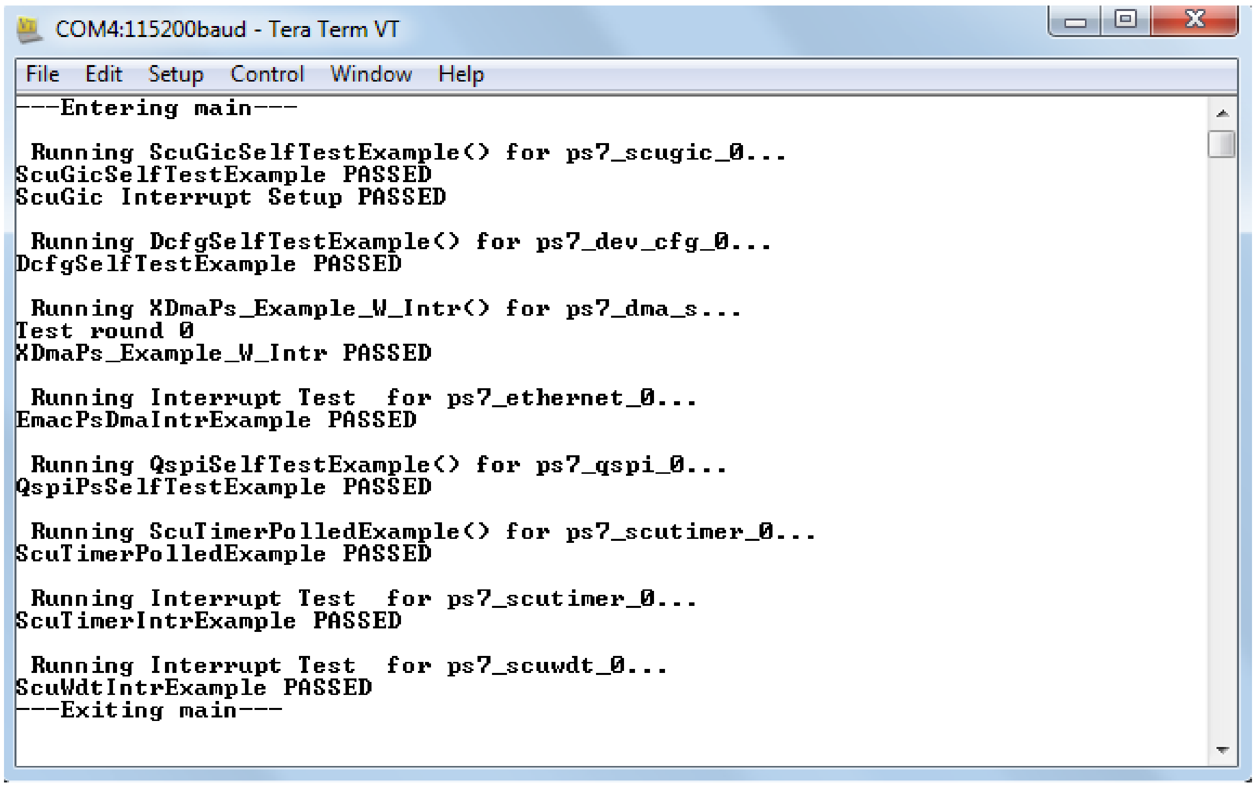


Figure 11 – PicoZed SOM SW1 Set to QSPI Boot 7010/20 on the Left; 7015/30 on the Right

9. Close or disconnect the terminal that may have previously been open on your PC.
10. Plug in the USB-UART cable.
11. *<ZedBoard & PicoZed Only>* Turn power on.
12. Launch a terminal program with the 115200/8/n/1/n settings.
13. Push the RST button. You should see the results in the terminal.


COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
---Entering main---

 Running ScuGicSelfTestExample() for ps7_scugic_0...
ScuGicSelfTestExample PASSED
ScuGic Interrupt Setup PASSED

 Running DcfgSelfTestExample() for ps7_dev_cfg_0...
DcfgSelfTestExample PASSED

 Running XDmaPs_Example_W_Intr() for ps7_dma_s...
Test round 0
XDmaPs_Example_W_Intr PASSED

 Running Interrupt Test  for ps7_ethernet_0...
EmacPsDmaIntrExample PASSED

 Running QspiSelfTestExample() for ps7_qspi_0...
QspiPsSelfTestExample PASSED

 Running ScuTimerPolledExample() for ps7_scutimer_0...
ScuTimerPolledExample PASSED

 Running Interrupt Test  for ps7_scutimer_0...
ScuTimerIntrExample PASSED

 Running Interrupt Test  for ps7_scuwdt_0...
ScuWdtIntrExample PASSED
---Exiting main---
```

Figure 2 – Results from QSPI boot of Periph_Test

14. Close the terminal.

15. *<ZedBoard & PicoZed Only>* Turn power off.

16. Unplug the USB-UART cable.

Experiment 3: Write and boot from the microSD Card

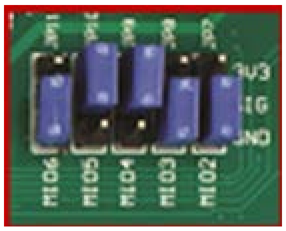
Next, we will prepare the microSD/SD card with the bin file that SDK generated. Then we will boot the same test application from the microSD/SD card.

Experiment 3 General Instruction:

Make a copy of the .bin file, naming it boot.bin, on the microSD/SD card. Change the MODE jumpers to SD boot. Insert the card and power on. Observe the results in the terminal.

Experiment 3 Step-by-Step Instructions:

1. *<MicroZed & PicoZed only>* To program a microSD Card, insert the microSD card into an adapter first.
2. Plug the SD Card or Adapter into your PC. Use Windows Explorer to browse to the `SDK_Workspace\Test_Peripherals\bootimage` folder. Copy the `Test_Peripherals.bin` image to the microSD/SD Card.
3. On the Card, rename the .bin file to **`boot.bin`**. THIS IS CRITICAL! The Stage 0 bootloader looks specifically for this file. Also, the name is not case sensitive.
4. Eject the card from the PC and insert it into the board's cage.
5. Set the Boot Jumpers to SD Card.
   a. ZedBoard and MicroZed



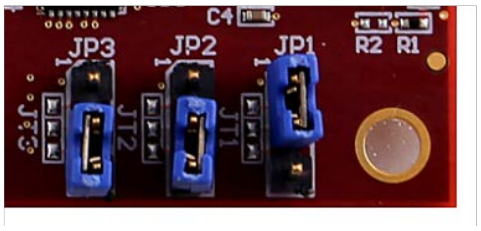


Figure 3 – SD Card Boot Mode: ZedBoard left; MicroZed right

b. PicoZed

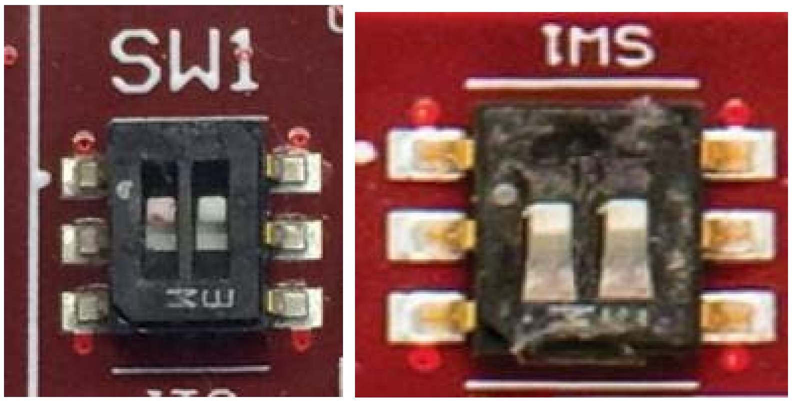


Figure 14 - PicoZed SW1 Set to SD Boot 7010/20 on the Left; 7015/30 on the Right

6. Close or disconnect the terminal that may have previously been open on your PC.
7. *<ZedBoard & PicoZed Only>* Turn power on.
8. Plug in the USB-UART cable.
9. Launch a terminal program with the 115200/8/n/1/n settings.
10. Push the RST button. You should see the same results in the terminal that were shown during the QSPI test.

Questions:

Answer the following questions:

- *Is the order of the images critical when generating the boot image? If so, list the order.*

____

- *True or False? The Create Zynq Boot Image tool creates both the QSPI and SD card images in one operation.*

____

- *What is the required name for the .bin file on the SD card when booting?*

____

This concludes Lab 7.

Revision History

Date	Version	Revision
12 Nov 13	01	Initial release
23 Nov 13	02	Revisions after pilot
01 May 14	03	MicroZed.org Training Course Release
10 Dec 14	04	Updated to Vivado 2014.3
07 Jan 15	05	Updated to Vivado 2014.4
12 Mar 15	06	Finalize SDK 2014.4
Oct 15	07	Updated to SDK 2015.2
Aug 16	08	Updated to SDK 2016.2

Resources

www.microzed.org

www.picozed.org

www.zedboard.org

www.xilinx.com/zynq

www.xilinx.com/sdk

www.xilinx.com/vivado

www.xilinx.com/support/documentation/sw_manuals/ug949-vivado-design-methodology.pdf

www.xilinx.com/support/documentation/sw_manuals/ug1046-ultrafast-design-methodology-guide.pdf

Answers

- *Is the order of the images critical when generating the boot image? If so, list the order.*

YES! FSBL ELF, then bitstream, then application ELF

- *True or False? The Create Zynq Boot Image tool creates both the QSPI and SD card images in one operation.*

False. SDK used to operate that way, but in 2014.3, it requires two separate operations.

- *What is the required name for the .bin file copied to the SD card?*

boot.bin